AMD DEVELOPER symposium 2002

# Optimizing for the AMD Opteron™ Processor

Tim Wilkens Ph.D.

# Agenda

AMD DEVELOPER symposium 2002

- Overview of AMD Opteron™ processor and x86-64
- Memory Optimizations
- Floating Point Optimizations / Performance
- Integer Optimizations / Performance
- Closing Points and Questions

# Introducing AMD Opteron™ Processor

AMD DEVELOPER symposium 2002

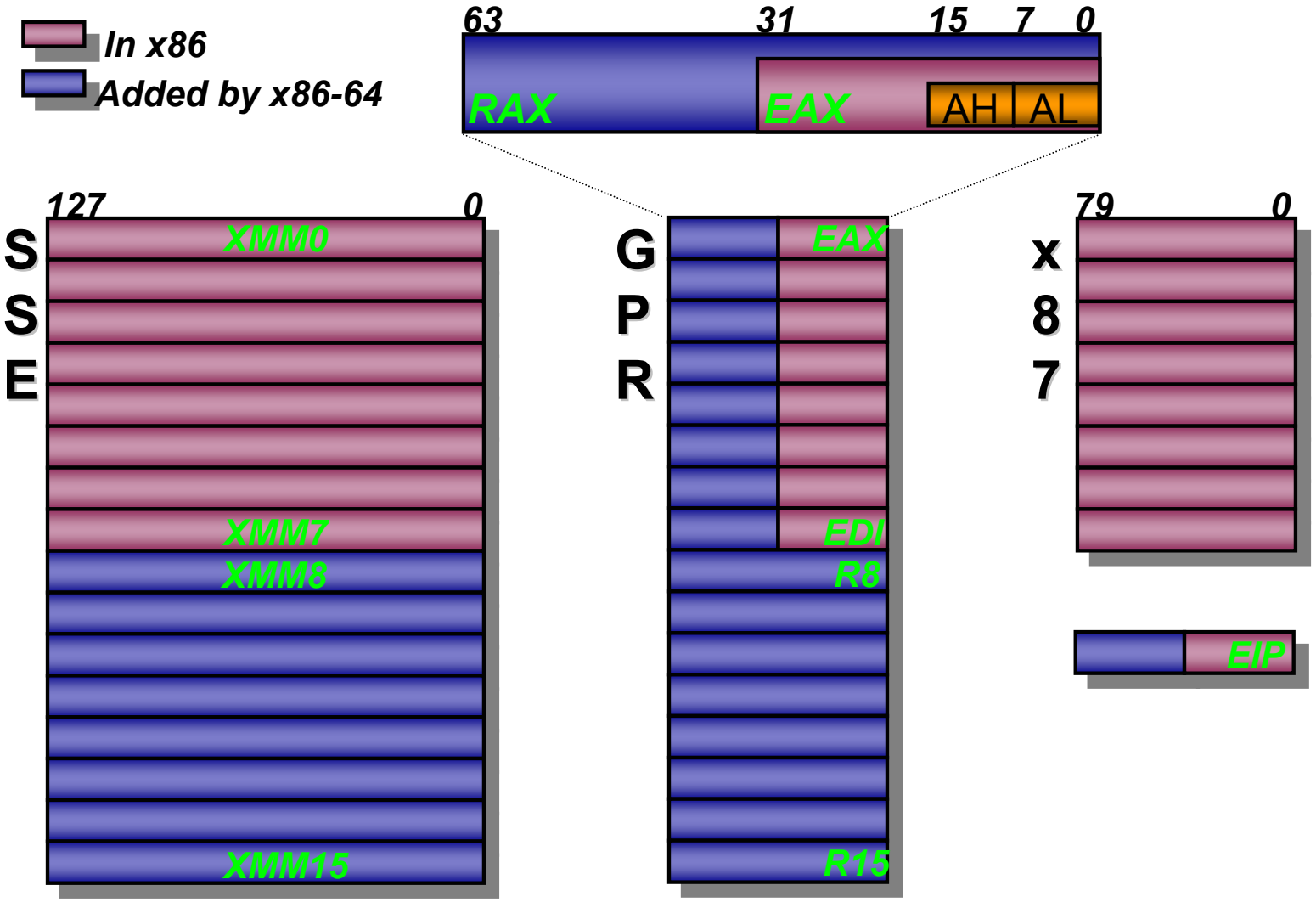
- ***32-Bit Compatibility***
  - Supports x86 assembly, MMX™ and 3DNow!™
  - *Supports* **SSE** and **SSE2**
    - ✓ in ***64-bit mode*** AMD Opteron™ processor has an *additional* 8 **SSE** / **SSE2** registers

    ***Check for SSE2 feature bit*** rather than just **CPU ID.**
    remember CPUID returns **"AuthenticAMD"** instead of **GenuineIntel"**

- ***New Features***
  - *Dedicated On Die Memory Controller designed to provide*
    - ✓ ***Up to 5.3*** **GB/s** of Bandwidth per processor
    - ✓ Less than ***80*** **ns** latency
  - *On Chip HyperTransport™ Links designed to provide*
    - ✓ Up to ***6.4*** **GB/s** of direct inter-processor bandwidth
    - ✓ Up to ***6.4*** **GB/s** of bandwidth to PCI-X and AGP 8x on each HT link

*….. and let's not forget,* ***AMD Opteron is a 64-bit processor***

# x86-64 Programmer's Model



# 64-bit Performance Enhancements

AMD DEVELOPER symposium 2002

- ***16*** **G**eneral **P**urpose **R**egisters (**GPR**s)
  - ***8*** standard **GPR**s (rax, rbx, rcx, rdx, rbp, rsp, rsi, rdi)
  - ***8*** additional 64-bit **GPR**s are available (***r8*** - ***r15***)
  - All **GPR**s addressable as *byte*, *word*, *dword* or *qword*

- ***2x*** as many **SSE** / **SSE2** registers
  - ***8 additional*** registers (***xmm8-xmm15***) *in addition to the standard* ***xmm0-xmm7***

***8 additional* GPRs** and **xmm** registers allow *developers* to ***reduce register pressure*** and ***more fully utilize the* FPU / ALU**

- ***x86-64* ABI**
  - INT function arguments passed via extra **GPR**s
  - FP function arguments passed via extra **xmm** registers
  - *Less pushing/popping* from memory required

# Memory Optimizations

# Prefetch Instructions

## (Lowering Apparent Memory Latency)

AMD DEVELOPER symposium 2002

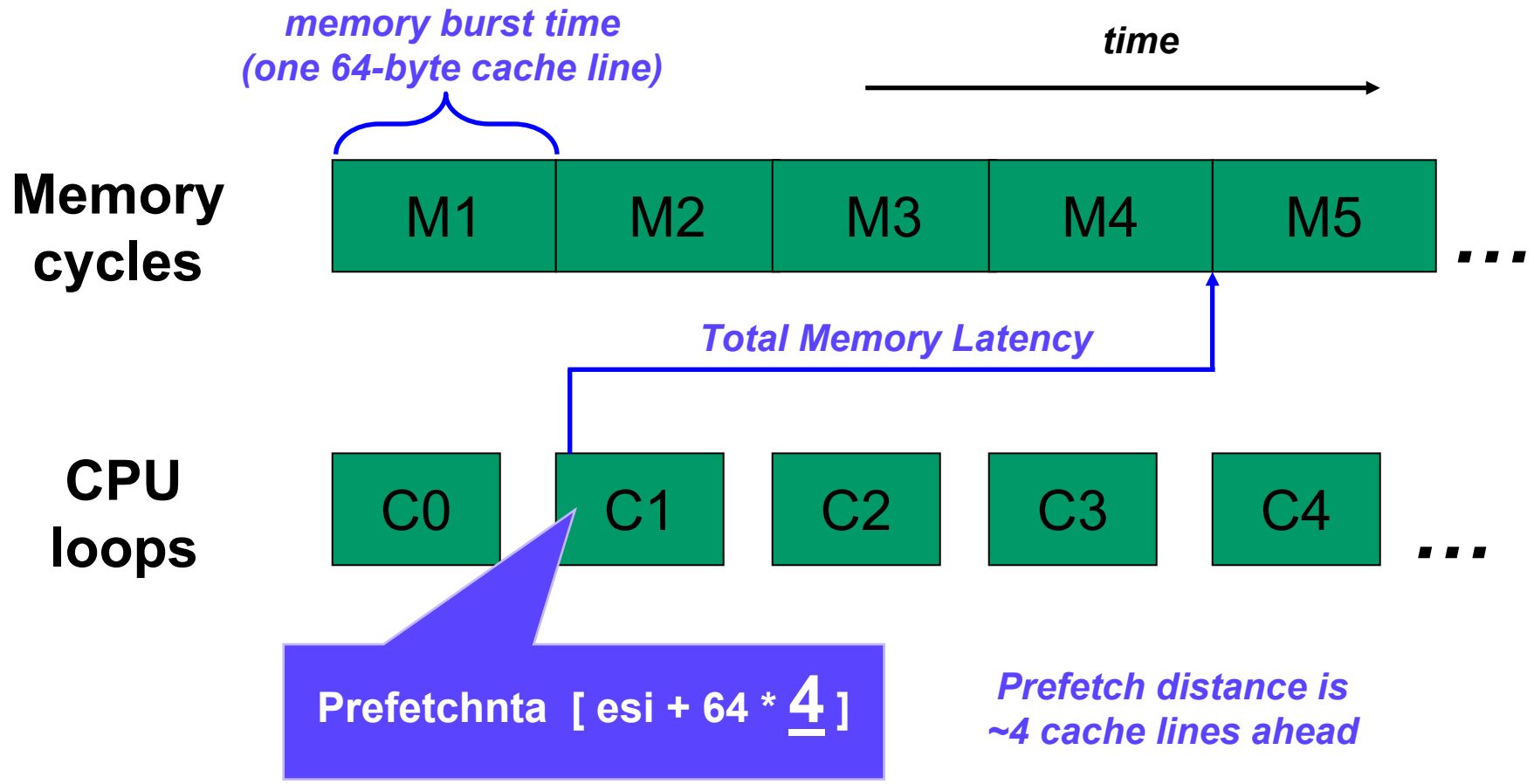
- Fetch data from memory **in advance of operating on it**
  - reduces apparent memory latency
- Prefetch Instructions are *always executed* by AMD **Opteron**™ processor
- 3 Prefetch Instructions
  - *prefetchnta*
    - ✓ Data moved into way 0 of L1
    - ✓ When data in L1 overwritten, data not evicted to L2
  - *prefetcht0*
    - ✓ Data moved into L1 and L2
  - *prefetcht1*
    - ✓ Data moved only into L2
- **Memory Bandwidth** can be increased by **2x*** that of *rep movsd*

  **performance includes the use of non-temporal writes to memory*

# Software Prefetch

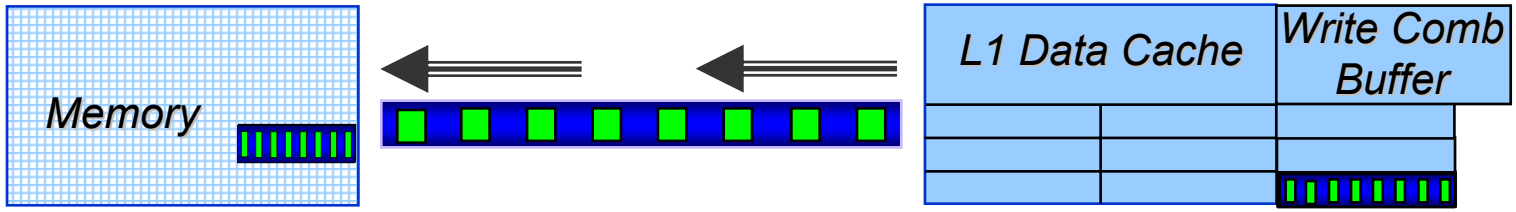
## Bandwidth Limited Case

AMD DEVELOPER symposium 2002



# Streaming Store
## (non-temporal write)






- Non-Streaming Store Memory Write
  - Disadvantages
    - ✓ ***Read** and **Write** of cacheline **performed***
    - ✓ **L1** / **L2** Caches are polluted

- Streaming Store Writes
  - Advantages
    - ✓ ***No** cacheline allocation (**reading**) **required***
    - ✓ **L1** / **L2** caches are not polluted
    - ✓ Chained writes to memory rather than read/write/read/write pattern

- Streaming store instructions (**movntq**, **movntdq**, **movntps**, **movntpd**)

# Data Alignment

AMD DEVELOPER symposium 2002

- **Align Data in SSE & SSE2 Code**
  - **Unaligned Loads/Stores**
    - ✓ movups and movupd are vector path
      - ***Only 2/3 of FPU and ALU pipes utilized***
      - *Long instruction latency – increases register pressure*
  - **Aligned Loads/Stores**
    - ✓ movaps and movapd are direct path doubles
      - ***All 3 pipelines in FPU or ALU continually fed***
      - *Latency of load is shorter – less register pressure*
- **Recommended Alignment**
  - *Double Words* (**4** Byte Aligned)  *Quad Words* (**8** Byte Aligned)

# Floating Point Optimizations

# Floating Point Unit

AMD DEVELOPER symposium 2002

- Features
  - ***36*** entry FPU instruction scheduler
  - **O**ut **O**f **O**rder (**OOO**) Execution
  - Support for x87, SSE, SSE2, and 3DNow!™ technology
  - ***16*** SSE & SSE2 128-bit xmm registers
  - ***8*** legacy x87 80-bit registers

- **FPU** Throughput
  - SSE2, x87
    - ✓ Theoretical: (**1** Mul + **1** Add)/cycle
    - ✓ Realized: 1.9 **FLOP**s/cycle
  - SSE, 3DNow!
    - ✓ Theoretical: (**2** Mul + **2** Add)/cycle
    - ✓ Realized: 3.4+ **FLOP**s/cycle

# SSE and SSE2

AMD DEVELOPER symposium 2002

- **SSE / SSE2** Floating Point Instructions
  - ***scalar***
    - ✓ *Operate* only *on **1** single or double* precision number
    - ✓ Compilers generate lots of scalar code
    - ✓ *Hammer executes scalar instructions efficiently*
    - ✓ Peak **SSE2** Scalar Performance 1.85+ FLOP/cycle
      (*No advantage to using packed instructions in double precision*)
  - ***packed***
    - ✓ *Operate* only *on **4** single or **2** double* precision numbers
    - ✓ Advantages
      - Better Code Density
      - **2x** *greater performance* **only** ***in single precision***

# Use Extra Registers
## (Loop Unrolling and Software Pipelining)

AMD DEVELOPER symposium 2002

How can more registers help?

- Loop Unrolling
  - ✓ Necessary to use additional registers
  - ✓ Increased utilization of *FPU and ALU pipelines*
- SSE and SSE2 Vector Instructions
  - ✓ **Vector Instructions have long latencies**
  - ✓ **More registers reduce register pressure**
  - ✓ Increased utilization of *FPU and ALU pipelines*
- *Fewer loads/stores to memory required*
  - ✓ New **ABI** reduces pushing and popping to and from memory

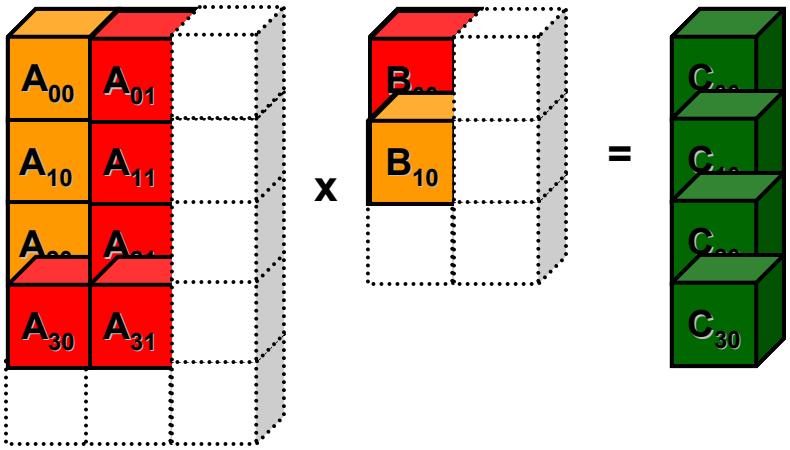
➢ Illustration:

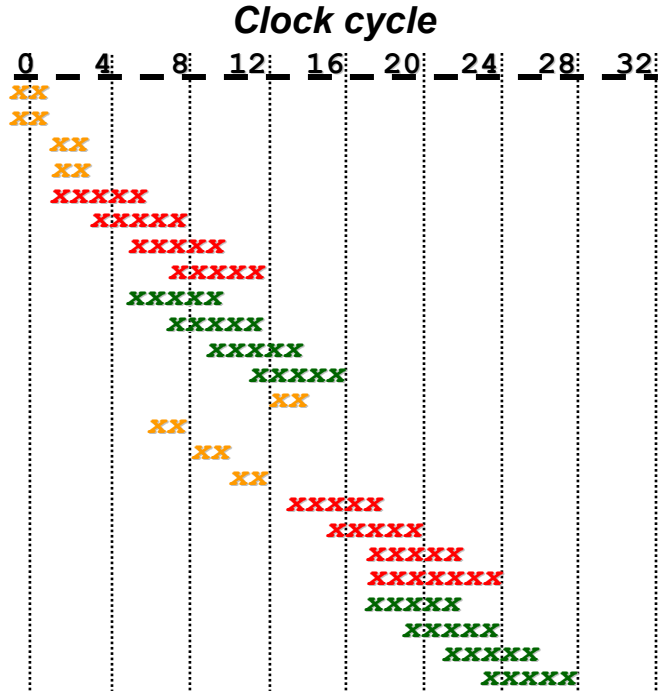
- **D**ouble Precision **GE**neral **M**atrix **M**ultiply (**DGEMM**) BLAS 3 Routine*

**determines in part the world's top 500 most powerful supercomputers*

# 32-Bit Software Pipelining
## (Optimized DGEMM Illustration)

AMD DEVELOPER symposium 2002



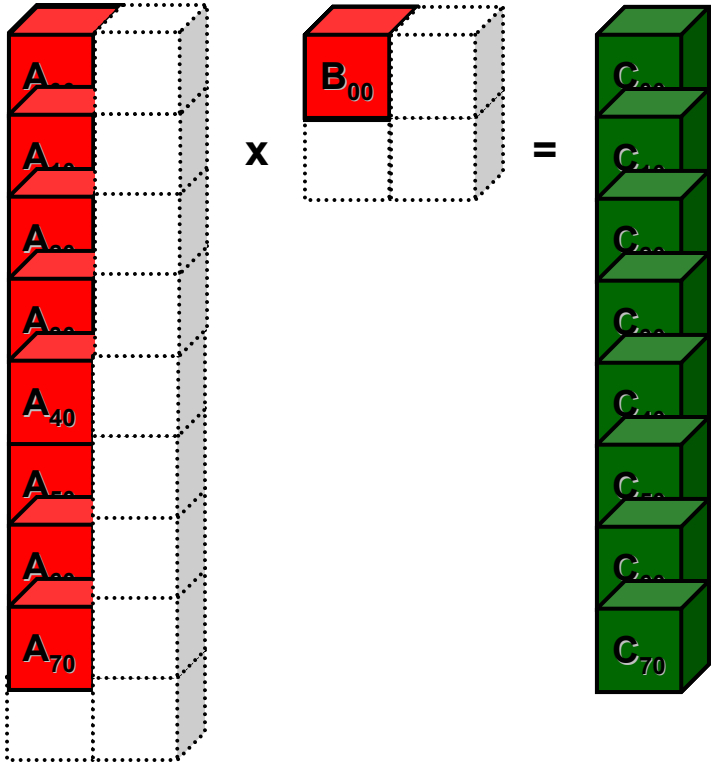

***SSE2 Packed Double Load***

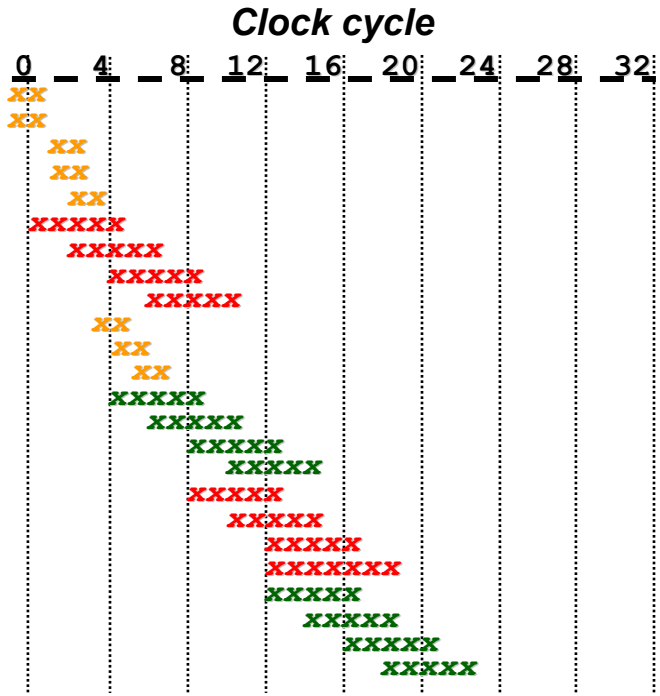
***SSE2 Packed Double Multiply***

***SSE2 Packed Double Add***

***29** Cycles Elapse*

# 64-Bit Software Pipelining
## (Optimized DGEMM Illustration)




***SSE2 Packed Double Load***

***SSE2 Packed Double Multiply***

***SSE2 Packed Double Add***

***24** Cycles Elapse*

# Maximize Prefetch Efficiency

## (Optimized Matrix Multiplication Illustration)

AMD DEVELOPER symposium 2002

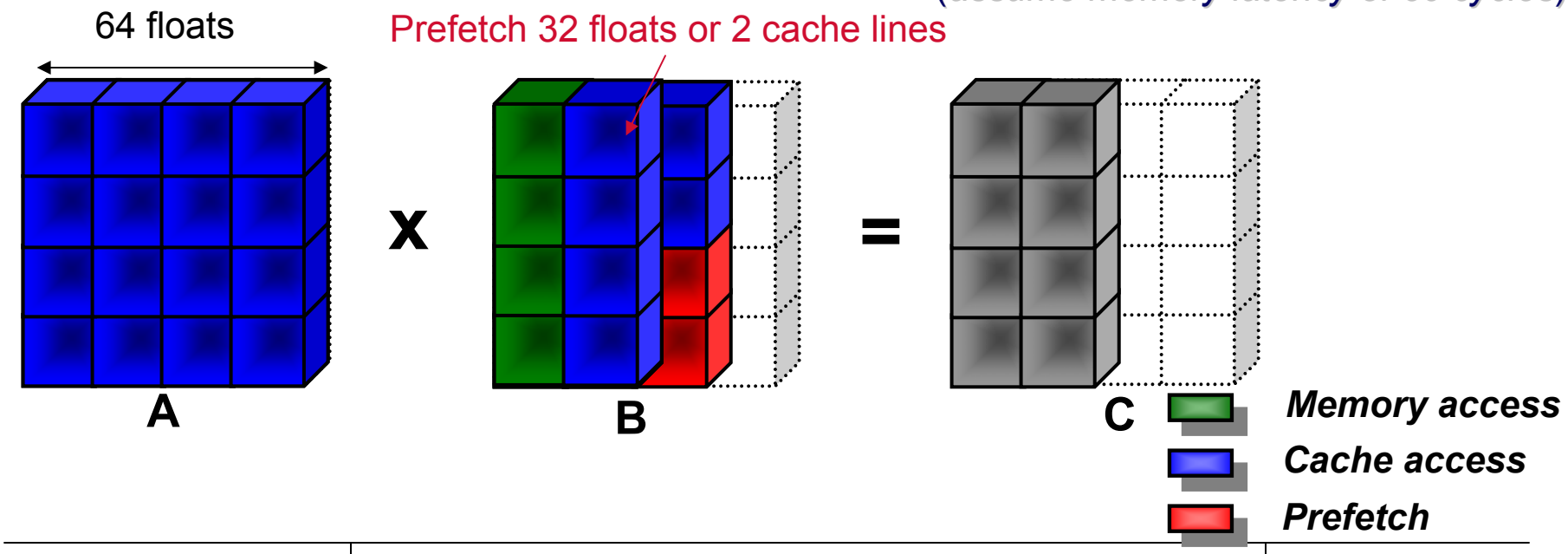
- Structure data (integer multiple of cache lines)

*Columns and Rows contain **64** float or **4** cachelines*

- Prefetch locations dictated by Memory Latency, Workload & Throughput

*Approximately **64** cycles expire every **2** dot products (**254** **FLOP**s)*
***254** **FLOP**s / **4** **FLOP**s Per Cycle = 64 cycles*

*(assume memory latency of **60** cycles)*



# Integer Optimizations

# Integer Unit

AMD DEVELOPER symposium 2002

- Features:
  - **32-bit**
    - ✓ 1 add / clock cycle
    - ✓ 1 multiply / clock cycle (***2x*** the throughput of AMD Athlon™ processor)
    - ✓ Multiply latency has shrunk from 5 cycles on AMD Athlon to 3 cycles
  - **64-bit**
    - ✓ 1 add / clock cycle
    - ✓ 1 multiply every other clock cycle
    - ✓ Multiply latency is 4 cycles
  - Integer Instruction Scheduler
    - ✓ **O**ut **O**f **O**rder (**OOO**) from a queue of 24* Integer Macro-Ops

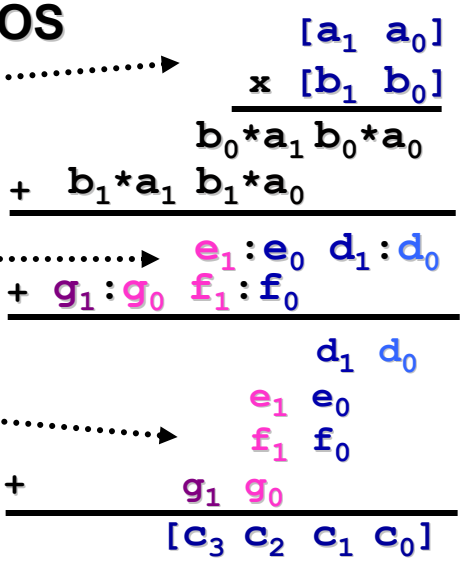
    **Athlon Instruction Scheduler was 18 Macro-Ops deep*

# Big Number Arithmetic

## (32-Bit and 64-Bit Integer Performance Evaluation)

AMD DEVELOPER symposium 2002

- Big Number Arithmetic is utilized in some encryption algorithms
- **128**, **256**, **512-bit** numbers represented as **32-bit** or **64-bit** *integer arrays*
  - $X_{128}^{32\text{-Bit}} = [x_3, x_2, x_1, x_0] = x_3 \cdot 2^{96} + x_2 \cdot 2^{64} + x_1 \cdot 2^{32} + x_0$
  - $X_{256}^{64\text{-Bit}} = [x_3, x_2, x_1, x_0] = x_3 \cdot 2^{192} + x_2 \cdot 2^{128} + x_1 \cdot 2^{64} + x_0$
- Illustration of Multiplying two **64-bit** numbers in ***x86* OS**
  - Multiply **64-bit** numbers `a` & `b`
  - Intermediate Products are *64-bits* stored in *edx*:*eax*
  - Accumulate column-wise with *8* **GPR**s using *adc* and *add* instructions
  - ***4*** integer multiplications and ***8*** additions required to compute **128-bit** result `c`



# Big Number Arithmetic

## (32-Bit and 64-Bit Integer Performance Evaluation)

AMD DEVELOPER symposium 2002

- Illustration of Multiplying two **64-bit** numbers in ***x86-64*** **OS**
  - Multiply **64-bit** numbers **a** & **b** ⋯⋯→ $[a_0]$ x $[b_0]$
  - Intermediate Products are *128-bits* stored in *rdx*:*rax* ⋯⋯→ $[c_1\ c_0]$
  - ***1*** integer multiplication required to compute **128-bit** result **c** in ***x86-64***

- Performance advantages of using **64-Bit** Arithmetic

| Big Number Size | *AMD Athlon™ processor* **32-Bit** *Performance* | *AMD Opteron™ processor* **32-Bit** *Performance* | *AMD Opteron* **64-Bit** *Performance* |
|---|---|---|---|
| 128-bits | ***100%*** | ***145%*** | ***529%*** |
| 256-bits | ***100%*** | ***144%*** | ***529%*** |
| 512-bits | ***100%*** | ***142%*** | ***569%*** |

# Closing Points

AMD DEVELOPER symposium 2002

- **Improve Code Performance**
  - ***Port to 64-Bit x86-64 OS***
    - ✓ Use 8 extra SSE / SSE2 XMM registers and 8 extra GPRs
    - ✓ Software Pipeline to effectively use ALU and FPU
  - ***Use Prefetch Instructions to maximize memory bandwidth***
  - ***Use SSE Packed instructions to maximize single precision performance***
  - ***Use Non-Temporal Streaming Store Instructions where appropriate***
  - ***Construct data storage structures to maximize locality to CPU***

# Trademark Attribution

AMD DEVELOPER symposium 2002

AMD, the AMD Arrow Logo, AMD Athlon, AMD Opteron and combinations thereof and 3DNow! are trademarks of Advanced Micro Devices, Inc. HyperTransport is a licensed trademark of the HyperTransport Technology Consortium. MMX is a trademark of Intel Corporation. Other product names used in this presentation are for identification purposes only and may be trademarks of their respective companies.

© 2002 Advanced Micro Devices, Inc.